WWW.SEMANTICSIMILARITY.ORG

SEMILAR API 1.0

User guide

7/29/2013

Outline

1. Introduction
2. Overview and FAQ
3. Data objects
4. Configurations
5. Preprocessing
6. Word to word similarity
7. Sentence to Sentence similarity
8. Document to document similarity
9. Corpus and Models
10. Using LDA tool
11. Citation information
12. References
13. Getting started with SEMILAR API

Introduction

The goal of the SEMantic simILARity software toolkit (SEMILAR; pronounced the same way as the word 'similar') is to promote productive, fair, and rigorous research advancements in the area of semantic similarity. Semantic similarity is a widely adopted approach to language understanding in which the meaning of a text A is inferred based on how similar it is to another text B, called benchmark text, who's meaning is known.

The SEMILAR software environment offers users, researchers, and developers, easy access to fully-implemented semantic similarity methods in one place through both a GUI-based interface and a library. Besides productivity advantages, SEMILAR provides a framework for the systematic comparison of various semantic similarity methods.

It should be noted that SEMILAR offers measures for computing the semantic similarity of texts at various levels of granularity: word-to-word, sentence-to-sentence, paragraph-to-paragraph, and document-to-document, or a combination of these such as sentence-to-document.

This document describes the SEMILAR library API. The GUI-based SEMILAR application is described in a separate document. This document presents concisely the various semantic similarity methods available in the SEMILAR library (Java) along with guidelines on how to use them.

Please visit http://www.semanticsimilarity.org/ for further details and recent updates. See the example codes and read me file for quick start. And feel free to contact us for any issues, suggestions you may have. Please find the details about references, attributions, and citation information in the later sections.

Dr. Vasile Rus (vrus@memphis.edu), Director.

Rajendra Banjade (rbanjade@memphis.edu)

Dr. Mihai Lintean (mclinten@memphis.edu)

Nobal Niraula (nbnraula@memphis.edu)

Dr. Dan Stefanescu (dstfnscu@memphis.edu)

Quick Overview

The following list of Frequently Asked Questions (FAQ) works as a quick overview of the SEMILAR API and is meant to save your time.

What are the similarity methods available in SEMILAR?

- SEMILAR API comes with various similarity methods based on Wordnet, Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), BLEU, Meteor, Pointwise Mutual Information (PMI), Dependency based methods, METEOR, optimized methods based on Quadratic Assignment, etc. Some methods have their own variations which coupled with parameter settings and your selection of preprocessing steps could result in a huge space of possible instances of the same basic method.

What is the granularity of similarity methods?

- SEMILAR contains methods to measure the semantic similarity at word level, i.e. word-to-word measures, sentence- to-sentence, paragraph-to-paragraph, and document-to-document. In addition, the methods can be applied to compute the semantic similarity of texts of different granularities such as word-to-sentence similarity, sentence-to-document similarity, etc.
- Please note that some methods expand word-to-word similarity measures to larger texts such as sentence-to-sentence whereas some methods are directly applicable between texts of any granularity. For example, there are variations of LDA-based similarity methods that work at word-level and others that can be used to compute sentence-level similarity. On the other hand, LSA can be directly used to compute the similarity of two words or two sentences (an LSA vector must be obtained for a sentence using vector algebra from the individual words' LSA vectors).

What are the word to word similarity methods?

- Please go to word to word similarity section. Note that there are some sentence level or larger text level similarity measures that are expanded from word-to-word similarity measures. Please pay attention using some similarity methods as they are backed by large data (for example, LSA, LDA, pairs of Wikipedia words and their PMI values make the huge file etc.). However, you can use them in separate runs as they may consume more memory when run together.

Where can I find the details about the methods/algorithms available in SEMILAR?

- This document only describes the API to implementations of the number of methods that address the task of semantic similarity. We do not have (yet) a single document describing in detail all the methods and algorithms available in SEMILAR. We offer a comprehensive list of

references to the original publications that introduced the various methods. Please find the reference section for more details.

Similarity and Relatedness are quite different things. Have you categorized them?

- Though similarity and relatedness are quite different concepts, we refer them as similarity in general. Some of the methods measure the similarities whereas others measure the relatedness. We should refer to their descriptions in details to characterize them. Corpus based models usually measure the relatedness.

I am not just doing word-to-word or sentence-to-sentence similarity research, my research is on text classification, clustering, text mining, information retrieval (or something related), machine translation evaluation, etc. How can I best utilize this tool?

- Certainly there are many ways of using word-to-word, sentence-to-sentence similarity or relatedness measures in information retrieval, text mining, clustering, classification, machine translation evaluation, etc. You may consider using word-to-word and/or sentence-to-sentence level similarity to document level or use document level similarity methods (work in progress) available in SEMILAR API itself.

What are the recent updates to the SEMILAR API?

- This document covers the methods available in SEMILAR API as of June 2013. Please visit http://www.semanticsimilarity.org for the details and most recent updates.

Which programming language is used to create SEMILAR API?

- Java (Jdk 1.7). And we plan to continue the development in this language.

How big is the SEMILAR package?

- It is a large library and application because it relies on large models and packages. Most of the NLP tools come with big models or other resources which are relatively are large (couple of hundred MBs) such as Standford/OpenNLP parser models, Wordnet lexical database, etc. In addition, our similarity methods also require pre-built models. For example, LSA spaces, LDA models, and Wikipedia PMI data are large components by themselves. If an user wants to utilize selective methods, there is no need to download everything. SEMILAR can be downloaded in separate zipped files for ease of customization and setup that fits various needs.

Which corpora or data sets are needed?

- We have generated LSA spaces and LDA models using TASA and the Wikipedia corpus whereas for PMI calculations we have used Wikipedia. These models can be downloaded from the SEMILAR website. The user may generate new models based on different corpora, with different preprocessing steps, or other settings. The Semilar API allows the user to specify new, non-

default model names and paths. Please see the corpora details section for more details about corpora we are using.

I want to generate and use LSA/LDA models using different corpora or requirements. Is it possible to generate and use my models in SEMILAR?

- Yes, you can develop LSA/LDA models using your own corpus. But you have to take care of the format of the model files, and certain file naming requirements to match your model name, etc. To ease your developing LSA/LDA models, we have provided an interface to the LSA (in progress), and LDA tools (please see the section Using LDA for more details on creating LDA models and the References section about the tools we are using). You may find it really helpful generating models using the SEMILAR API as the formatting of the output matches format used by the other SEMILAR components.

What is done during preprocessing and what tools are available?

- Some similarity methods require certain kinds of preprocessing such as POS tagging, parsing, stemming/lemmatization, etc. Tokenization and removing punctuations is needed by all methods. The SEMILAR preprocessor has options to select Standford tools or OpenNLP tools for tokenization, tagging, and parsing. For stemming, you can select Porter's stemmer, or WordNet based stemmer (the latter guarantees the stem is a proper word).

Can I skip preprocessing or do it myself?

- You may preprocess your texts without using the SEMILAR preprocessor but your responsibility would be to create certain objects and populate corresponding field values in these objects.
- You may not need to do any preprocessing or just do the basics to use your selected methods. Please check the preprocessing requirements for particular methods you may want to use.

How much time consuming methods are there in SEMILAR?

- It depends. Most of the methods are quite fast. Some optimization methods that also rely on syntactic or other types of information may be slower.

How much memory is consumed?

- It depends on the particular method. Most of the implemented methods should work well on regular desktop and laptop computers.

Which Wordnet version is used?

- Wordnet 3.0 as of June 2013.

Can SEMILAR be used for languages other than English?

- Similarity measures that are available in SEMILAR have been developed with English in mind and there are no models included for languages other than English. But it's possible to adapt the

methods to other languages. For instance, you can generate LSA/LDA models using texts from a target, other-than-English language (remember that you can develop LSA/LDA models using interface/functions available in SEMILAR API itself) and then use SEMILAR LSA and LDA similarity measures to compute the similarity of texts in the target language.

Where can I find references and citation information?

- Please find below the sections - SEMILAR Citation info and References for more details.

What are the licensing terms of using SEMILAR?

- Its free to use for non-commercial, academic and research purposes. Please note that we provide the licensing and information about third-party components that are being used in SEMILAR in the reference section in on the website at http://www.semanticsimilarity.org. Complying with the SEMILAR licensing terms implies complying with license agreements issued by the third parties for the components included in SEMILAR. Please read the license agreement first before downloading and installing SEMILAR.

Are there any examples for quick start using it?

- Yes. Example code is available in SEMILAR package in the extracted SEMILAR folder. There are different methods, options, possible values of parameters, different models you can choose from etc. You have to go through this guide and latest information from the website to make best use of SEMILAR.

I want to run all methods at once, is it possible?

- It's possible. Some of the preprocessing tools and similarity methods have to load huge models in memory. You can probably run all other methods at once if your machine has at least 8 GB of memory.

What platforms (OS) does it support?

- SEMILAR API can be used in Linux and Windows. Jdk 1.7 or higher is required.

I have encountered some error or exception, how can I diagnose?

- We try to provide as much support as possible. Please provide enough details when you encounter any issues. It's possible that errors can be caused by missing required files, misplaced folders (especially after extracting the zipped files), misspelling, incorrect input format, corrupted download file, etc.

What is the similarity score if the given word(s) is not in the vocabulary? For example: Your LSA model may not have some of the words I would like to see the similarity score for.

- We are scrutinizing all the methods to make sure that user will not get confused in odd situations (for example: user gives a word pair and gets similarity score zero just because they are not available in the model. Does it mean they are actually not similar?). Many of the similarity measures give you back a very odd number (-9999). In this case, user should understand that it was not possible to calculate the similarity for the given word pairs.

I have little background on one or more similarity methods. Rather than description of methods, can I find solid steps (or examples) to get the results?

- Well, we understand that sometimes just knowing the theory doesn't make you comfortable using tools such as SEMILAR because of configurations, different similar looking functions and so on. To make it easy (as much as possible) to use with less effort, we have different resources including this manual. It is not possible to document every detail, so please find the example codes along with the SEMILAR main package, check for other resources, and let us know if you get into trouble.

I have few questions, issues about using it, can I get some help?

- Yes sure. Please feel free to write to Rajendra Banjade (rbanjade@memphis.edu) and Dr. Vasile Rus (vrus@memphis.edu).

Data Objects

The following are the data objects you should be familiar with. Please note that it doesn't provide trivial details.

Word:

Class	semilar.data.Word
Description	This object represents a word (or token).
Fields	Raw form – Without any preprocessing, as given by the user. Base form – stemmed/lemmatized form. POS – part of speech tag. isStopWord – is it a stop word? isPunctuation – is it a punctuation?
Enumeration	NA
Methods	NA

Sentence:

Class	semilar.data.Sentence
Description	This class represents a sentence.
Fields	Raw form – Without any preprocessing, as given by the user. Words – List containing the list of Words. Syntactic tree –Syntactic tree (string form). Dependencies – dependency information. Dependency tree – dependency tree (string form).
Enumeration	NA
Methods	Getter/setters

Note: The details about the document representation will be published soon.

Configurations

The required files and location where to put the downloadable resources are available at SEMILAR website**.** To avoid any potential problems, please create a specific folder for SEMILAR and organize all the data in their default locations as specified in the SEMILAR website so that you can save some time for your actual work. However, if you really need it, you can organize the resources differently and specify the locations using configuration manager described below.

If you want to put data files – Open NLP, Standford, Wordnet, or LSA/LDA/PMI data files at some folders other than the default locations (as mentioned in the SEMILAR website), you can set the file/folder paths using static methods of configuration manager class. For example,

semilar.config.ConfigManager
ConfigManager.setSemilarHomeFolder(String path) – set the SEMILAR home folder (i.e. folder containing the SEMILAR API jar file. It is essentially the home folder of project that uses the SEMILAR API). ConfigManager.setSemilarDataRootFolder(String path) – set the SEMILAR data root folder. It includes LSA/LDA models, test data set etc. ConfigManager. setWordnetPath(String path) – set the Wordnet root folder (end with /). ConfigManager. setLsaRootFolder (String path) – folder where LSA models are kept. The LSA model files should be in the folder (in the LSA root folder) named as the model name. For example, TASA.

Preprocessing

During preprocessing, you provide word, sentence, or larger text as input in the raw form and preprocessor processes and creates an object representing the given input. For example, when you provide sentence as input, then the preprocessor creates an object of class 'Sentence'.

If you want to do preprocessing outside, then your responsibility would be to populate the data object(s) described above. Please see the description of methods you want to use and their preprocessing requirements. But creating the same representation is quite tricky, so we recommend you using the preprocessor of SEMILAR itself. Please see the example codes for the usage details.

Word to word similarity/relatedness methods

All word to word similarity methods have implemented functions computeWordSimilarity(Word word1, Word word2) – requires POS tag and computeWordSimilarityNoPos(Word w1, Word w2) – doesn't require POS tag. And they return the similarity score in the range of 0 to 1.

Please see the word to word similarity example codes available in the SEMILAR website and find the papers from the reference section for more details.

Sentence to Sentence similarity methods

Please note that there are basically two ways to calculate sentence to sentence similarity. One is to expand word to word similarity (i.e. use similarity of word in one sentence to a word in another sentence and by some means calculate sentence level similarity score) and another approach is to have semantic representation of sentence and use that to calculate the sentence similarity directly. All of the word to word similarity methods described above can be expanded for sentence level similarity. However, it's not always possible to semantically represent the sentence directly such as in Wordnet, we have to deal at word level. But there are cases where you have choice whether to use word to word similarity or use sentence level representation to measure sentence similarity. For example, using LSA for semantic representation, either semantic representation of sentence (by adding the word vectors) can be used to calculate sentence similarity or expand word (using semantic representation of individual word) level similarity to sentence level without explicitly representing the sentence.

For example, optimal matching solution for sentence to sentence similarity is based on word-to-word similarity measures. The optimal lexical matching is based on the optimal assignment problem, a fundamental combinatorial optimization problem which consists of finding a maximum weight matching in a weighted bipartite graph. Whereas, dependency based method requires word to word similarity as well as some grammatical relations.

The similarity methods return the score in the range of 0 to 1. However, the range within 0 to 1 can vary from method to method.

Please see the sentence to sentence similarity examples available in the SEMILAR website.

Document to document similarity

We are adding the similarity functions for bigger texts. As of July 2013, the similarity method available in the SEMILAR is LDA based method. Using LDA for similarity is a two steps process, one is to infer the probability distributions of documents over topics based on some LDA model and use those distributions to calculate the similarity of documents. Please see the example codes and inline comments for further details.

Corpus and Models

TASA corpus (one of the very popular corpus): LSA and LDA models are developed from the lemmatized TASA corpus.

English Wikipedia articles (Jan 2013 Snapshot): LSA models and PMI values calculated from Wiki texts. Please contact us if you want to use the clean Wikipedia texts and PMI data. These are not available for the download from SEMILAR website as their size is quite large (8GB+).

Using LDA tool

We have provided an interface to LDA tool (Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi, 2008). A TASA LDA model is available in SEMILAR website for download but to measure the similarity of sentences or larger text, you have to infer document distributions over topics. However, for word to word similarity we use the TASA LDA model by default. So the sentence to sentence similarity expanding the word to word is also possible without inferring the document distribution over topics.

So, if you want to measure the document (or sentence) similarity using LDA except expanding word level similarity, you have to have LDA model and have to infer the probability distributions for your documents. You may use the LDA models already available in SEMILAR and just infer probability distributions for your documents based on the already available models, OR generate the LDA models from your text collection and do the inferencing.

Below is some description of interface to the LDA tool. Please see the example codes and inline comments for better understanding on how to use it.

LDA Estimator:

If you just want to infer the probability distributions using the already developed LDA model (LDA model generated using TASA corpus is available in SEMILAR website), please skip this and go to the section: LDA Inferencing (below).

Class	semilar.tools.lda.LDAEstimator
Description	This is a wrapper around the JGibbLDA tool (Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi, 2008) for estimating document distribution over topics and topic distribution over the words in the vocabulary. First line of the input file should contain the number of documents in that file. Document means anything that is in the single line. So, if you want to create LDA model using documents containing multiple lines, make a single line for that document. Please find the example code and input/output files in the similar package.
Parameters	LDA data folder – folder where the input data file exists. The output will be generated in the same folder. Input file name – Name of the input data file. The first line should contain the number of documents in that file. Model name – name the LDA model. Note that the output files will be created with this name, and during inferencing (please see below), this name will be used while loading model for the inferencing.

	Number of topics, Alpha, Beta, Number of iterations, words per topic – please see LDA documentations.
Function	startEstimation() – starts estimation.
Dependencies	NA
Preprocessing	Preprocess the input text as you like before providing it to the LDA for modeling.
Input data	File containing the documents. The first line should contain the number of documents in that file. Please find the sample input file for LDA estimator available the SEMILAR package.
Output	Model files in the lda data folder specified above. The output files are, <model-name>.info – information about the model such as number of topics etc. <model-name>.phi - topic distribution over words. <model-name>.theta – document distribution over topics. <model-name>.twords – topics <model-name>.wordmap – Vocabulary along with their frequencies.

LDA Inferencing:

Class	semilar.tools.lda.LDAInferencer
Description	This is a wrapper around the JGibbLDA tool (Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi, 2008) for inferencing document distribution over topics and topic distribution over the words in the vocabulary. Please find the example code and input/output files in the similar package.
Parameters	LDA data folder – folder where the input data file and model files generated during estimation exist (please see LDA estimation). The output will be generated in the same folder. Input file name – Name of the input data file. Model name – name the LDA model. Note that the model name should match with the name of the model you estimated.
Function	startInferencing () – starts inferencing.
Dependencies	NA
Preprocessing	Preprocess the input text as you like.
Input data	File containing the documents. First line of the input file should contain the number of documents in that file. Document means anything that is in the single line. So, if you want to create LDA model using documents containing multiple lines, make a single line for that document. Model files – The output generated during estimation (please see above). The model name should match. **Note**: If you want to use the TASA model and do infer those probability distributions for your input data, then keep the TASA model files (please find the LDA model files and their default location from similar website) to the LDA directory you specified and give the model name “TASA”.

Output	Model files in the lda data folder specified above. The output files are, <input file name>.<model-name>.info – information about the model such as number of topics etc. <input file name>.<model-name>.phi - Topic distributions over words. <input file name>.<model-name>.theta – Document distribution over topics <input file name>.<model-name>.twords – Topics <input file name>.<model-name>.wordmap – Words and their frequencies in the corpus.

SEMILAR Citation info

Rus, V., Lintean, M., Banjade, R., Niraula, N., and Stefanescu, D. (2013). SEMILAR: The Semantic Similarity Toolkit. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, August 4-9, 2013, Sofia, Bulgaria.

References

Rus, Vasile, Nobal Niraula, and Rajendra Banjade. "Similarity Measures Based on Latent Dirichlet Allocation." *Computational Linguistics and Intelligent Text Processing*. Springer Berlin Heidelberg, 2013. 459-470.

Rus, Vasile, Lintean Mihai. "A Comparison of Greedy and Optimal Assessment of Natural Language Student Input Using Word-to-Word Similarity Metrics"

Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). WordNet::Similarity -Measuring the Relatedness of Concepts, *In the Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pp. 1024-1025, July 25-29, 2004, San Jose, CA (Intelligent Systems Demonstration).

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. *In Proc. of The 17th International World Wide Web Conference (WWW 2008)*, pp.91-100, April 2008, Beijing, China.

Corley, C. and Mihalcea, R. (2005). Measuring the semantic similarity of texts. In *Proceedings of ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*. [1, 15, 34].

Michael Denkowski and Alon Lavie, "Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems", *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*, 2011.

Papineni, Kishore, et al. "BLEU: a method for automatic evaluation of machine translation." *Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics*, 2002.

Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. "An introduction to latent semantic analysis." *Discourse processes* 25.2-3 (1998): 259-284.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *the Journal of machine Learning research* 3 (2003): 993-1022.

Gabrilovich, Evgeniy, and Shaul Markovitch. "Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis." *IJCAI*. Vol. 7. 2007.

Church, Kenneth Ward, and Patrick Hanks. "Word association norms, mutual information, and lexicography." *Computational linguistics* 16.1 (1990): 22-29.

Please continue reading for the details on getting started with SEMILAR API

Getting started with SEMILAR API 1.0

NOTE:

1. Please see the user agreement file first (it should be available in the SEMILAR download package or you may find it from the semanticsimilarity.org).
2. Please see the introductory sections of this user manual. You will find the overview of SEMILAR and is meant to save your time.
3. Since, examples help a lot to quick start using SEMILAR, please find them in the SEMILAR package or download from the website. Please see the details below on how to make them work.
4. You may use some selected methods and you may not need to download some of the data files based on your selection. Please find the detail from the SEMILAR website. Some files are very big, for instance, cleaned Wikipedia text, PMI values. Those files are not available in download. Please email us to get them.
5. Please visit http://semanticsimilarity.org for the recent updates.
6. If you have any issues, questions, suggestions or need some help, please feel free to contact Rajendra Banjade (rbanjade@memphis.edu) and Dr. Vasile Rus (vrus@memphis.edu).

Prerequisites

1. Jdk 1.7 or higher.
2. OS: Windows/Linux.
3. Can be run on regular workstation. However, running different methods together may be quite heavy.

A: Steps for downloading SEMILAR package and Creating test project

1. Please create a Java project from Eclipse or Netbeans. Say SemilarDemo and let's call the project home folder as SEMILAR home.
2. Download the SEMILAR main package and extract in the SEMILAR home folder. So, the SEMILAR project home folder will contain SEMILAR API jar file and some dependent files.
3. Download the example code from the SEMILAR website. Extract the zip file and add those example code files in your project. You may need to fix the package name in the example code files to match with the package name you imported them into.

4. Compile errors? Add the SEMILAR API into the Classpath (i.e. add the SEMILAR jar into your project). SEMILAR API should be in the SEMILAR home folder.

B: Downloading and setting up data files

1. Create a folder at some place in your file system, say Semilar-data. We will call it as SEMILAR data home folder (please note that SEMILAR home folder is the demo project home folder, as described above. But this is the SEMILAR data home folder).
2. Download LSA model files. Extract it and put the LSA-MODELS folder in the SEMILAR data home folder.
3. Similar to LSA, download the LDA model files. Extract it and put the LDA-MODELS folder in the data home folder.
4. Download and extract the Word to word similarity test dataset (Word2Word-Similarity-test-data). And put that in the data folder.
5. Download and extract the LDA tool test dataset (LDA-tool-test-data). And put that in the data folder.

C: Running example codes

1. Please read the comments in the code.
2. Once you read the summary below, please see the code for the details.
3. Word2WordSimilarityTest.java file contains the demonstration code for word to word similarity. Comment/uncomment some of the word metrics, as running them altogether may be quite heavy (it depends on the machine you are using). Please set the SEMILAR data home folder in the code.
   For example: ConfigManager.setSemilarDataRootFolder("C:/Users/<user name>/data/Semilar-data/");
4. Sentence2SentenceSimilarityTest.java file has example code for sentence to sentence similarity. Comment/uncomment some of the methods and run the file. Running all of the methods together may not work if you are using regular machine. Similar to the word to word similarity test file, set the SEMILAR data home folder. Please note that for METEOR method, you have to provide the project home folder.
5. LDABasedDocumentSimilarityTest.java file shows how to measure the similarity of documents using LDA based method. Please note that document may be a single sentence or bigger text, but we refer them as document especially while working with LDA. Measuring similarity of documents using LDA is somewhat different from other methods, so we created a separate example code file.
6. LDATest.java file contains the example code showing how to use LDA tool to estimate the probability distributions and infer them for the new documents based on the already available LDA model. Please see the details about the LDA tool and reference about it in the SEMILAR API guide available in the website or available with the SEMILAR package.

7. Got errors? Usually the source of errors are missing files, extracting the files in the wrong place, providing the wrong data path etc. If you really got into trouble, please feel free to contact us.
8. Once you are able to run all or some of the methods, you may try changing parameters or try using different models etc.